Chapter 2: C++ Getting Started

Contents

Time required: 120 minutes

Read: Think C++

- Chapter 2 Variables and Types

Do: Online Tutorials

- C++ Tutorial
- C++ Intro
- C++ Syntax
- C++ Output (Print Text)
- C++ New Lines
- C++ Comments

- C++ Variables
    - C++ Declare Multiple Variables
    - C++ Identifiers
    - C++ Constants
- C++ User Input
- C++ Data Types
    - C++ Numbers
    - C++ Booleans
    - C++ Characters
    - C++ Strings
- C++ Operators
    - C++ Assignment Operators
    - C++ Comparison Operators
    - C++ Logical Operators

Introduction to C++

C++ (pronounced see plus plus) is a general-purpose programming language that is freeform and compiled to a binary executable. It is regarded as an intermediate-level language, as it comprises both high-level and low-level language features. It provides imperative, object-oriented, and generic programming features.

Machine Independent but Platform Dependent: A C++ executable is not platform-independent (compiled programs on Linux won't run on Windows), however they are machine independent. A C++ program can be compiled and executed on any operating system that has a C++ compiler.

C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms. As an efficient performance driven programming language, it is used in systems software, application software, device drivers, embedded software, high-performance server and client applications, and entertainment software such as video games. Various entities provide both open source and proprietary C++ compiler software, including the FSF, LLVM, Microsoft and Intel.

What is C++?

C++ is a cross-platform language that can be used to create high-performance applications.

Who developed C++?

Bjarne Stroustrup developed C++ (originally named "C with Classes") in 1983 at Bell Labs as an enhancement to the C programming language.

What Is a Program?

A program is a sequence of instructions that specifies how to perform a computation. The computation might be something mathematical, like solving a system of equations or finding the roots of a polynomial. It can also be a symbolic computation, like searching and replacing text in a document or (strangely enough) compiling a program.

The instructions (or commands, or statements) look different in different programming languages, but there are a few basic functions that appear in just about every language:

input: Get data from the keyboard, or a file, or some other device.

output: Display data on the screen or send data to a file or other device.

math: Perform basic mathematical operations like addition and multiplication.

testing: Check for certain conditions and execute the appropriate statements.

repetition: Perform some action repeatedly, usually with some variation.

Believe it or not, that's pretty much all there is to it. Every program you've ever used, no matter how complicated, is made up of functions that look more or less like these. Thus, one way to describe programming is the process of breaking a large, complex task up into smaller and smaller subtasks until eventually the subtasks are simple enough to be performed with one of these simple functions.

Tutorial 1: Hello World

Traditionally the first program people write in a new language is called "Hello, World." because all it does is print the words "Hello, World." In C++, this program looks like the following.

Create a C++ program file named: **hello_world.cpp** Enter the following code

```cpp
//============================================================================
// Name        : hello_world.cpp
// Author      :
// Version     :
// Description : Hello World in C++
//============================================================================

// Include iostream for the cout function
#include <iostream>

// Entry point for all C++ programs
int main()
{
    // std:: is the standard library
    // Use cout to print text to the console
    // endl adds a new line
    std::cout << "Hello, World!" << std::endl;
    std::cout << "We are on the C++ road to programming excellence!" << std::endl;
    return 0;
}
```

Example run:

```
Hello, World!
We are on the C++ road to programming excellence!
```

How It Works

1. `// Your first C++ program`

   In C++, any line starting with **//** is a comment. Comments are intended for the person reading the code to better understand the functionality of the program. It is completely ignored by the C++ compiler.

2. `#include <iostream>`

   The **#include** statement is a preprocessor directive used to include files in our program. The above code includes the contents of the **iostream** library file. A library file is prewritten code that we can use without having to write it ourselves.

   This allows us to use **cout** and **endl** in our program to print output and create a new line on the screen.

3. `int main() {...}`

   A valid C++ program must have the main() function. The curly braces indicate the start and the end of the function.

The execution of code starts in this function. **`std::cout << "Hello World!";`**

`std::cout` prints the content inside the quotation marks to the default output device. This is typically the console or screen. It must be followed by the extraction operator **<<** followed by a string. In our example, "Hello World!" is the string.

Note: A semicolon (;) is used to indicate the end of a statement.

4. `return 0;`

The **`return 0;`** statement is the "Exit status" of the program. 0 indicates a successful run.

In published C++ code you sometimes will see a statement such as the following:

```
std::cout << 4 << std::endl;
```

This statement behaves exactly like the following statement:

```
std::cout << 4 << "\n";
```

Values and Variables

The number four (4) is an example of a numeric value. In mathematics, 4 is an integer value. Integers are whole numbers, which means they have no fractional parts, and an integer can be positive, negative, or zero. Examples of integers include 4, -19, 0, and -1005. In contrast, 4.5 is not an integer, since it is not a whole number.

Floating-point numbers are an approximation of mathematical real numbers. As in the case of the int data type, the range of floating-point numbers is limited, since each value requires a fixed amount of memory. In some ways, though, ints are very different from doubles. Any integer within the range of the int data type can be represented exactly. This is not true for the floating-point types. Consider the real number pi. Since pi contains an infinite number of digits, a floating-point number with finite precision can only approximate its value.

Variables in C++ must be defined before they can be used.

```
int number;
double aDoubleVariable = 24.0;
float aSecond{60.0}
```

The following table lists some of the data types in C++.

Type	Size (in bytes)	Range

short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
signed char	1	-128 to 127
unsigned char	1	0 to 255
Float	4	7 decimal points
Double	8	15 decimal points

Expressions

- **statement**: a unit of code that does something – a basic building block of a program.
- **expression:** a statement that has a value – for instance, a number, a string, the sum of two numbers, etc. 4+2, x-1, and "Hello, world!\n" are all expressions.

Input

User-entered values are passed to the input stream using `cin` which takes input from the keyboard using the operator <<.

```
#include <iostream>
#include <iostream>
int main()
{
    double degrees;
    double celsius;
    std::cout << "Enter temperature in degrees F: ";
    // Program pauses until use enters a value and presses enter
    std::cin >> degrees;
    celsius = (degrees - 32) * 5.0 / 9.0;
    std::cout << degrees << "° Fahrenheit = " << celsius << "° Celsius" <<
std::endl;
    return 0;
}
```

Tutorial 2: Favorite Number

Create a C++ program named **favorite_number.cpp**

```cpp
/**
 * Filename: favorite_number.cpp
 * Written by:
 * Written on:
 * Revised:
 * Get number from user
 */

#include <iostream>

int main()
{
    // Declare variable
    int favoriteNumber = 0;
    // Output to console
    std::cout << "\nEnter your favorite number between 1 and 100: ";
    // Get input from keyboard
    std::cin >> favoriteNumber;
    // Output to console
    std::cout << "Amazing! " << favoriteNumber << " is my favorite number too.\n";
    return 0;
}
```

Example run:

```
Enter your favorite number between 1 and 100: 42
Amazing! 42 is my favorite number too.
```

Constants

C++ supports named constants. Constants are declared like variables with the addition of the const keyword:

```
const double pi = 3.14159265358979323846;
```

This is the standard representation of pi in C++ as a constant.

Once declared and initialized, a constant can be used like a variable in all but one way—a constant may not be reassigned. It is illegal for a constant to appear on the left side of the assignment operator (=) outside its declaration statement. A subsequent statement like

```
pi = 2.5;
```

would cause the compiler to issue an error message.

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.